Convergence analysis for column-action methods in image reconstruction

Elfving, Tommy; Hansen, Per Christian; Nikazad, Touraj

Published in:
Numerical Algorithms

Link to article, DOI:
10.1007/s11075-016-0232-6

Publication date:
2016

Document Version
Peer reviewed version

Numerical Algorithms manuscript No.
(will be inserted by the editor)

Convergence Analysis for Column-Action Methods in Image Reconstruction

Tommy Elfving · Per Christian Hansen · Touraj Nikazad

Received: date / Accepted: date

Abstract Column-oriented versions of algebraic iterative methods are interesting alternatives to their row-version counterparts: they converge to a least squares solution, and they provide a basis for saving computational work by skipping small updates. In this paper we consider the case of noise-free data. We present a convergence analysis of the column algorithms, we discuss two techniques (loping and flagging) for reducing the work, and we establish some convergence results for methods that utilize these techniques. The performance of the algorithms is illustrated with numerical examples from computed tomography.

Keywords Algebraic iterative reconstruction · Block-iteration · ART · Kaczmarz · Cimmino · Convergence

Mathematics Subject Classification (2000) 65F10 · 65R32

1 Introduction

Filtered back projection, FDK and similar "direct" reconstruction methods in computed tomography give excellent results when we have plenty of data and when the noise in the data is low. But for situations with high noise and/or limited data (which

This work is a part of the project HD-Tomo funded by Advanced Grant No. 291405 from the European Research Council.

Tommy Elfving
Department of Mathematics, Linköping University, SE-581 83 Linköping, Sweden
E-mail: toelf@mai.liu.se

Per Christian Hansen
Department of Applied Mathematics and Computer Science, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark
E-mail: pcha@dtu.dk

Touraj Nikazad
School of Mathematics, Iran University of Science and Technology, Narmak, Tehran, 16846-13114, Iran
E-mail: tnikazad@iust.ac.ir

arise, e.g., when one needs to limit the X-ray dose), or when it is desirable to incorporate constraints, an algebraic approach is often preferred.

Censor [6] coined the expression "row-action methods" for a specific class of algebraic iterative methods. This class includes Kaczmarz's algorithm, which was independently suggested under the name "ART" in [15] where it was used for the first time in the open literature to solve tomographic reconstruction problems.

It is also known [4], [13] how to base an iterative reconstruction algorithm on *columns* rather than on rows. The main advantage of the column version is that it does not exhibit the cyclic convergence of the row version, but converges to a least squares solution. Another advantage is the possibility for saving computational work during the iterations, as explained in Section 4.

Column-oriented algorithms have not been explored much in the literature. An exception is Watt [23] who derives a column-based reconstruction method and compares it with ART (also using nonnegativity constraints). A more recent paper is [3] where a two-parameter algorithm based on a block-column partitioning is studied. For specific choices of the parameters the block-SOR method (to be defined in Section 3) is obtained. Convergence results are given for the case where the coefficient matrix has full column rank.

Here we will study column-based iterations both theoretically and experimentally. Our starting point is a large linear system of equations,

$$A\,x \approx b, \qquad A \in \mathbb{R}^{m\times n}. \tag{1}$$

We make no assumptions about the dimensions or rank of A, and the system is not assumed to be consistent. The systems we have in mind typically arise from discretization of ill-posed problems such as computed tomography, and our numerical examples come from such applications. Since this paper deals with classic asymptotic convergence analysis we assume that there is no noise in the data.

The row and column methods seek to solve different problems. The row methods aim to compute a minimum-norm solution to a consistent system of equations, while the column methods aim to compute a least squares solution. Hence, for inconsistent problems the asymptotic behavior of the methods is different. The row-action methods exhibit cyclic convergence [11] but not in general to a least squares solution. The column methods, on the other hand, converge to a least squares solution but not in general to the minimum norm solution.

We will here only consider cyclic control, i.e., the rows/columns are selected in a cyclic order. For other controls see, e.g., [11, p.80], [22].

Our paper is organized as follows. In Section 2 we set the stage by briefly summarizing the well-known row action methods. Section 3 discusses the column versions, and we derive and discuss the associated convergence properties. In Section 4 we demonstrate how computational work can be reduced by not performing small updates, typically of solution elements that have converged. Section 5 summarizes our work. Numerical examples are given in the relevant sections of the paper. The connection of the column methods to optimization algorithms is briefly discussed in Appendix A, and in Appendix B we argue why A^TA is a dense matrix in computed tomography.

We use the following notation: $\rho(A)$ denotes spectral radius of a matrix A, $\mathcal{R}(A)$ and $\mathcal{N}(A)$ denote the range (column space) and null space of A, respectively, $A^\dagger$ denotes the Moore-Penrose pseudoinverse of A, and $P_{\mathcal{S}}$ denotes the orthogonal projector on the subspace $\mathcal{S}$. Moreover, if F_i are matrices and g_{ii} are the diagonal elements of a

matrix G, then we define

$$\text{blockdiag}(F_i) = \begin{pmatrix} F_1 & & \\ & F_2 & \\ & & \ddots \end{pmatrix}, \qquad \text{diag}(G) = \begin{pmatrix} g_{11} & & \\ & g_{22} & \\ & & \ddots \end{pmatrix}. \tag{2}$$

2 Sequential Block-Row Iteration

To set the stage for our analysis we first briefly summarize main results for the row-oriented method. Let A be partitioned into p disjoint block rows and let b be partitioned accordingly,

$$A = \begin{pmatrix} B_1 \\ \vdots \\ B_p \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_p \end{pmatrix}, \qquad B_i \in \mathbb{R}^{m_i \times n}, \quad b_i \in \mathbb{R}^{m_i}, \quad i = 1, \ldots, p.$$

Also, let $\{\omega_i\}_{i=1}^p$ be a set of positive relaxation parameters and let $M_i \in \mathbb{R}^{m_i \times m_i}$, $i = 1, 2, \ldots, p$ be a set of given symmetric positive definite (spd) matrices. The following generic algorithm, which uses the blocks B_i in a *sequential* fashion, covers several important special cases.

Algorithm BRI: Block-Row Iteration

Initialization: $x^0 \in \mathbb{R}^n$ is arbitrary.
For $k = 0, 1, 2, \ldots$ (cycles or outer iterations)
  $z^0 = x^k$
  For $i = 1, 2, \ldots, p$ (inner iterations)
    $z^i = z^{i-1} + \omega_i B_i^T M_i \big(b_i - B_i z^{i-1}\big)$
  End
  $x^{k+1} = z^p$
End

This method was considered in [13] for a special choice of the weight matrix M_i, and for general M_i in [12]. With $p = 1$ there is just one block and the method becomes fully simultaneous. And when $p = m$ each block consists of a single row so $M_i \in \mathbb{R}$, $i = 1, 2, \ldots, m$, and the iteration becomes fully sequential.

Let a *cycle* denote one pass through all blocks, i.e., one outer iteration. Since block-row iteration uses a single block in every inner iteration it takes p inner iterations to complete a cycle. One cycle of the above algorithm can then be written as (cf. [14], [21, p. 155])

$$x^{k+1} = x^k + A^T M_{\mathrm{r}}(A)\,(b - A\,x^k) = W_{\mathrm{r}}(A)\,x^k + A^T M_{\mathrm{r}}(A)\,b, \tag{3}$$

where

$$W_{\mathrm{r}}(A) = I - A^T M_{\mathrm{r}}(A)\,A, \quad M_{\mathrm{r}}(A) = (D_{\mathrm{r}} + L_{\mathrm{r}})^{-1}, \quad D_{\mathrm{r}} = \text{blockdiag}(\omega_i M_i)^{-1} \tag{4}$$

(the latter being block diagonal), and L_{r} is the block-lower triangular matrix

$$L_{\mathrm{r}} = \begin{pmatrix} 0 & & & \\ B_2 B_1^T & 0 & & \\ \vdots & \ddots & \ddots & \\ B_p B_1^T & \dots & B_p B_{p-1}^T & 0 \end{pmatrix}. \tag{5}$$

The following convergence result is from [12] and [14]. Assume that $0 < \epsilon < 2$. If

$$\omega_i \in \big(\epsilon, (2-\epsilon)/\rho(B_i^T M_i B_i)\big), \quad i = 1, 2, \dots, p, \tag{6}$$

then the iteration (3) converges towards a solution of

$$A^T M_{\mathrm{r}}(A)\,(b - Ax) = 0. \tag{7}$$

The convergence of Algorithm BRI was studied in [21] in an infinite dimensional Hilbert space setting.

Several well known iterative methods arise as special cases. The "block-Kaczmarz" method [13] uses $M_i = (B_i B_i^T)^{-1}$, so that $B_i^T M_i B_i = P_{\mathcal{R}(B_i)}$, the orthogonal projector onto the range of B_i. It follows that $\|B_i^T M_i B_i\|_2 = 1$ and hence we find, by (6), the well-known result that the method converges for $\omega_i \in (0, 2)$.

A second example is obtained with the diagonal matrix $M_i = \mathrm{diag}(B_i^T B_i)^{-1}$. For $p = 1$ we obtain the (relaxed) Cimmino method with $M_i = M = \mathrm{diag}(B^T B)^{-1}$. It is easy to see that $\|B_i^T M_i B_i\|_2 \le 1$ so that again convergence occurs for $\omega_i \in (0, 2)$. However the upper bound 2 is quite restrictive ((6) is only a sufficient condition for convergence) especially for large and sparse matrices, so that taking too small value of ω_i may result in poor rate of initial convergence. The methods BICAV [10] and DROP [9] were constructed to improve the rate by explicitly allowing the relaxation parameters to depend on sparsity. For a fully dense matrix the three methods coincide.

As a final example we mention SART [2]. The methods SART, BICAV and Cimmino are also treated in [8, section 7] both for linear equations and linear inequalities.

3 Sequential Block-Column Iteration

We now consider the column version of the above algorithm and its convergence properties. Our discussion is formulated entirely in a linear algebra setting, but we wish to emphasize that column iterations are closely connected to coordinate-descent optimization algorithms, and we refer to Appendix A for the optimization perspective.

3.1 The Generic Column-Oriented Algorithm

Let A be partitioned into q disjoint block columns and let x be partitioned accordingly,

$$A = \begin{pmatrix} A_1 & A_2 & \cdots & A_q \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_q \end{pmatrix}, \qquad A_i \in \mathbb{R}^{m \times n_i}, \quad x_i \in \mathbb{R}^{n_i}, \quad i = 1, 2, \dots, q.$$

Moreover let $\{\omega_i\}_{i=1}^q$ be a set of positive relaxation parameters, and let $M_i \in \mathbb{R}^{n_i \times n_i}$, $i = 1, 2, \ldots, q$ be a set of given spd matrices. The following generic algorithm, which uses the blocks A_i in a *sequential* way, provides a generic framework for column-oriented iterations.

Algorithm BCI: Block-Column Iteration

Initialization: $x^0 \in \mathbb{R}^n$ is arbitrary; $r^{0,1} = b - A\,x^0$.
For $k = 0, 1, 2, \ldots$ (cycles or outer iterations)
  For $i = 1, 2, \ldots, q$ (inner iterations)
    $x_i^{k+1} = x_i^k + \omega_i M_i A_i^T r^{k,i}$
    $r^{k,i+1} = r^{k,i} - A_i(x_i^{k+1} - x_i^k)$
  End
  $r^{k+1,1} = r^{k,q+1}$
End

Note that the residual updating in the inner iteration of the above algorithm is an efficient way to compute the residual given by

$$r^{k,i+1} = b - \sum_{j=1}^{i} A_j x_j^{k+1} - \sum_{j=i+1}^{q} A_j x_j^k. \tag{8}$$

Hence for each cycle the method requires one application of A_i and A_i^T for $i = 1, 2, \ldots, q$, and the total associated work corresponds to one multiplication with A and one multiplication with A^T. We will now derive an expression for one cycle of the algorithm.

Proposition 1 *One cycle of Algorithm BCI can be written*

$$x^{k+1} = x^k + M_{\rm c}(A)\,A^T(b - A\,x^k) = W_{\rm c}(A)\,x^k + M_{\rm c}(A)\,A^T b, \tag{9}$$

where

$$W_{\rm c}(A) = I - M_{\rm c}(A)\,A^T A, \quad M_{\rm c}(A) = (D_{\rm c} + L_{\rm c})^{-1}, \quad D_{\rm c} = \text{blockdiag}(\omega_i M_i)^{-1}, \tag{10}$$

and $L_{\rm c}$ is the block-lower triangular matrix

$$L_{\rm c} = \begin{pmatrix} 0 & & & \\ A_2^T A_1 & 0 & & \\ \vdots & \ddots & \ddots & \\ A_p^T A_1 & \ldots & A_q^T A_{q-1} & 0 \end{pmatrix}. \tag{11}$$

Proof For ease of notation we put $\omega_i = \omega$, $i = 1, 2, \ldots, q$ and within this proof we also define $\widetilde{D}_{\rm c} = \text{blockdiag}(M_i^{-1})$. Then (9) can be written

$$(\widetilde{D}_{\rm c} + \omega L_{\rm c})\,x^{k+1} = (\widetilde{D}_{\rm c} + \omega L_{\rm c})\,x^k + \omega A^T(b - Ax^k). \tag{12}$$

Using that $A^T A = L_{\rm c} + \bar{D}_{\rm c} + L_{\rm c}^T$ with $\bar{D}_{\rm c} = \text{blockdiag}(A_i^T A_i)$, we can rewrite the right hand side of Eq. (12) as

$$\big(\widetilde{D}_{\rm c} - \omega L_{\rm c}^T - \omega \bar{D}_{\rm c} + \omega A^T A\big)x^k + \omega A^T(b - Ax^k) = \big(\widetilde{D}_{\rm c} - \omega L_{\rm c}^T - \omega \bar{D}_{\rm c}\big)x^k + \omega A^T b.$$

Hence the ith block-component of (12) becomes

$$M_i^{-1}x_i^{k+1} = M_i^{-1}x_i^k - \omega\Big(A_i^T A_i x_i^k + (L_{\rm c}^T x^k)_i + (L_{\rm c}x^{k+1})_i - A_i^T b\Big),$$

which can be rewritten as

$$x_i^{k+1} = x_i^k + \omega\, M_i\, A_i^T\left(b - \sum_{j=1}^{i-1} A_j x_j^{k+1} - \sum_{j=i}^{q} A_j x_j^k\right),$$

which equals the ith step of the inner loop. The extension to the case of varying relaxation parameters $\{\omega_i\}$ is straight forward. □

Corollary 1 *Assume that the iterates of (9) converge to some limit x^+. Then*

$$A^T A\, x^+ = A^T b, \tag{13}$$

i.e., x^+ is a least squares solution of (1).

Proof The result follows by taking limits in (9) and noting that $M_{\rm c}(A)$ is nonsingular. □

Remark 1 The iterates in (9) are not sensitive to the ordering of the equations since, with a permutation matrix Π, we have $(\Pi A)^T(\Pi A) = A^T A$. The iterates do, however, depend on the ordering of the unknowns. Since row-iteration (implicitly) works with AA^T the opposite is true for these methods.

3.2 Convergence Analysis

We have seen that there exist several convergent members of the row-iteration scheme depending on the choice of weight matrices $\{M_i\}$. One may ask if corresponding members also converge using column-iteration? The answer is in the affirmative as we will show below. In our first analysis we assume that $\mathcal{N}(A) = \emptyset$. We discuss this case explicitly since it reveals the close connection between the row and column methods (cf. [13, lemma 3]).

It is instructive to compare the iterates produced by column-iteration and row-iteration, respectively, on two similar linear systems. Assume that column-iteration, using a specific set of relaxation parameters $\{\omega_i\}$, weight matrices $\{M_i\}$, and column-partitioning, is applied to (1). The resulting iteration matrix (using Proposition 1) becomes

$$W_{\rm c}(A) = I - M_{\rm c}(A)A^T A, \tag{14}$$

and we have that

$$x^k - x^+ = W_{\rm c}(A)^k(x^0 - x^+), \tag{15}$$

where x^+ is a least squares solution of (1).

Next assume that row-iteration is applied to a system of the form $A^T y = d$ using the same set of relaxation parameters, weight matrices, and block partitioning (which now are block rows of A^T). Then by construction

$$M_{\rm c}(A) = M_{\rm r}(A^T).$$

Further by (4) we have

$$W_{\mathrm{r}}(A^T) = I - AM_{\mathrm{r}}(A^T)A^T. \tag{16}$$

We recall that the matrix products AB and BA have the same nonzero eigenvalues. It follows with $\lambda_{\mathrm{e}}(\cdot)$ the set of eigenvalues excluding $+1$,

$$\lambda_{\mathrm{e}}\big(W_{\mathrm{c}}(A)\big) = \lambda_{\mathrm{e}}\big(I - AM_{\mathrm{c}}(A)A^T\big) = \lambda_{\mathrm{e}}\big(I - AM_{\mathrm{r}}(A^T)A^T\big) = \lambda_{\mathrm{e}}\big(W_{\mathrm{r}}(A^T)\big).$$

By choosing the relaxation parameters and weight matrices such that the row-iteration defines a convergent method it holds that $|\lambda_{\mathrm{e}}(W_{\mathrm{r}}(A^T))| < 1$. Next we consider the case $\lambda(W_{\mathrm{c}}(A)) = +1$. Then by (14) and the fact that $M_{\mathrm{c}}(A)$ is nonsingular and that $\mathcal{N}(A)$ and $\mathcal{R}(A^T)$ are orthogonal subspaces it holds

$$W_{\mathrm{c}}(A)\, v = v \quad \Leftrightarrow \quad v \in \mathcal{N}(A). \tag{17}$$

We thus arrive at the following result, with $\epsilon \in (0, 2)$:

Proposition 2 *Assume that $\mathcal{N}(A) = \emptyset$ and that $\omega_i \in \big(\epsilon, (2-\epsilon)/\rho(A_i M_i A_i^T)\big)$, $i = 1, 2, \ldots, q$. Then the sequence generated by Algorithm BCI converges to a least squares solution.*

Proof By the above reasoning, and conditions (6), with B_i replaced by A_i^T it follows that $\lim_{k\to\infty} W_{\mathrm{c}}(A)^k \to 0$. Using (15) the result follows. □

In order to treat the rank-deficient case we will use the following classical result by Keller [20, corollary 2.1] (adapted to our notation; recall that a matrix W is convergent if $\lim_{k\to\infty} W^k$ exists).

Theorem 1 *The iteration matrix $W_{\mathrm{c}}(A) = I - M_{\mathrm{c}}(A)\, A^T A$ is convergent if and only if the matrix $2D_{\mathrm{c}} - \mathrm{blockdiag}(A_i^T A_i)$ is positive definite.*

Proposition 3 *The iterates of Algorithm BCI converge to a solution of (13) if for any $0 < \epsilon < 2$ it holds*

$$\omega_i \in \big(\epsilon, (2-\epsilon)/\rho(A_i M_i A_i^T)\big), \quad i = 1, 2, \ldots, q. \tag{18}$$

Proof We will check when the matrix $\Delta = 2D_{\mathrm{c}} - \mathrm{blockdiag}(A_i^T A_i)$ is positive definite. Since both D_{c} and $\mathrm{blockdiag}(A_i^T A_i)$ are block-diagonal with the same size of the blocks it suffices to investigate the inequality (with $v_i \in \mathbb{R}^{n_i}$, $v_i \neq 0$)

$$\frac{2}{\omega_i} {v_i}^T M_i^{-1} v_i - {v_i}^T A_i^T A_i\, v_i > 0, \quad i = 1, 2, \ldots, q$$

or equivalently

$$0 < \omega_i < 2/c_i, \quad c_i = \frac{{v_i}^T A_i^T A_i v_i}{v_i M_i^{-1} v_i}.$$

Put $v_i = M_i^{1/2} \xi_i$. Then

$$c_i = \big({\xi_i}^T M_i^{1/2} A_i^T A_i M_i^{1/2} \xi_i\big)/\|\xi_i\|_2^2 = \big\|A_i M_i^{1/2} \xi_i\big\|_2^2 / \|\xi_i\|_2^2 \leq \big\|A_i M_i^{1/2}\big\|_2^2.$$

Hence if $\omega_i \in \big(0, 2/\|A_i M_i^{1/2}\|_2^2\big)$ then Δ is positive definite. Now

$$\big\|A_i M_i^{1/2}\big\|_2^2 = \big\|M_i^{1/2} A_i^T A_i M_i^{1/2}\big\|_2 = \rho\big((M_i^{1/2} A_i^T)(A_i M_i^{1/2})\big) = \rho\big(A_i M_i A_i^T\big),$$

which verifies (18). If $\rho(W_c(A)) < 1$ then the limit matrix is zero, and the result follows using (15). When $\rho(W_c(A)) = 1$, and since $W_c(A)$ is convergent, the limit of $W_c(A)^k$ is the projector onto $\mathcal{N}(I - W_c(A))$ along $\mathcal{R}(I - W_c(A))$. (This is in fact the orthogonal projector onto $\mathcal{N}(I - W_c(A)$; this follows easily since $\text{fix}(W_c(A)) = \text{fix}(W_c(A)^T) = \mathcal{N}(A)$, where fix$(\cdot)$ is the set of fixed points). Now $\mathcal{N}(I - W_c(A)) = \mathcal{N}(M_c(A)A^T A) = \mathcal{N}(A)$. It follows by (15) that $x^k \to x^+ + z$ with $z \in \mathcal{N}(A)$. Hence the limit vector satisfies the normal equations (13). □

We can now make the following conclusions regarding the convergence of the algorithms. In the consistent case and assuming $x^0 \in \mathcal{R}(A^T)$, the iterates of algorithm BRI converge towards the unique solution of minimum norm, whereas the iterates of algorithm BCI converge to a solution of (1) but not necessarily the one of minimal norm. In the inconsistent case there is also a difference: The iterates of algorithm BRI exhibit cyclic convergence, see [11], while the iterates of algorithm BCI converge to a least squares solution as shown in Proposition 3. In this section we only consider stationary iterations, but we remark that using a special type of iteration-dependent relaxation parameters $\{\omega_k\}$ the iterates of Algorithm BCI do converge to a weighted least squares solution. These parameters should fulfill $\omega_k \to 0$ and $\sum_{k=0}^{\infty} \omega_k = 0$, see [7] and [19].

3.3 Some Special Cases

In this section we take a look at certain special cases of Algorithm BCI. Regarding the choice of q:

$q = 1$. In this case $n_i = n$ and we have $\omega_i = \omega$ and $M_i = M \in \mathbb{R}^{n\times n}$. The iteration is *fully simultaneous* and takes the following form with x^0 given and $r^0 = b - A\,x^0$:

$$\left.\begin{aligned} x^{k+1} &= x^k + \omega M A^T r^k \\ r^{k+1} &= r^k + A(x^{k+1} - x^k) \end{aligned}\right\} \qquad k = 0, 1, 2, \ldots$$

$1 < q < n$. In this case the method is a *sequential block-column method*.

$q = n$. In this case $n_i = 1$ and $M_i \in \mathbb{R}$ (a scalar) for $i = 1, 2, \ldots, n$. This is called a *fully sequential* (or point) method.

We next discuss the **SOR**-case where we define

$$M_i = (A_i^T A_i)^\dagger \in \mathbb{R}^{n_i \times n_i}, \quad i = 1, 2, \ldots, q.$$

One way to implement this method is to solve a sequence of least squares problems:

$$\begin{aligned} y_i &= \text{argmin}_y \|A_i y - r^{k,i}\|_2, \\ x_i^{k+1} &= x_i^k + \omega_i y_i, \\ r^{k,i+1} &= r^{k,i} + A_i(x_i^{k+1} - x_i^k). \end{aligned}$$

If $A_i^T A_i$ is sparse or $n_i \ll n$ it may be more efficient to compute M_i or factorizations of $A_i^T A_i$ once and for all. We now investigate the condition (18):

$$\rho(A_i M_i A_i^T) = \left\|A_i M_i A_i^T\right\|_2 = \left\|A_i (A_i^T A_i)^\dagger A_i^T\right\|_2 = \left\|A_i A_i^\dagger\right\|_2 = 1.$$

So condition (18) becomes (as expected for SOR)

$$\omega_i \in (\epsilon, 2-\epsilon), \quad i = 1, 2, \ldots, q.$$

In some applications, such as computed tomography, it is very common that $A_i^T A_i$ is a full matrix; see Appendix B for an explanation of this. Hence it is relevant to consider an alternative to the SOR-case, namely, the **Cimmino**-case. Let a_i^j denote the jth column of block A_i and define

$$M_i = \frac{1}{n_i}\left(\operatorname{diag}\left(A_i^T A_i\right)\right)^{-1} = \frac{1}{n_i}\operatorname{blockdiag}\left(\|a_i^j\|_2^{-2}\right), \quad j = 1, 2, \ldots, n_i. \tag{19}$$

Note that the matrix can be considered as using a diagonal approximation of the matrix from SOR. We again investigate condition (18). It holds

$$\begin{aligned}\rho\left(A_i M_i A_i^T\right) &= \left\|A_i M_i A_i^T\right\|_2 = \frac{1}{n_i}\left\|\sum_{j=1}^{n_i} \frac{1}{\|a_i^j\|_2^2} a_i^j (a_i^j)^T\right\|_2 \\ &= \frac{1}{n_i}\left\|\sum_{j=1}^{n_i} P_{\mathcal{R}(a_i^j)}\right\|_2 \leq 1 \quad \Rightarrow \quad \omega_i \in (0, 2).\end{aligned} \tag{20}$$

We remark however that now the upper bound 2 is only a sufficient condition (similarly as for row-iteration) and it may lead to slow rate of convergence.

Now consider **BICAV**. Let s_i^ν be the number of nonzero elements in row ν of A_i, and let

$$M_i = \begin{pmatrix} \|a_i^1\|_{S_i} & & \\ & \|a_i^2\|_{S_i} & \\ & & \ddots \end{pmatrix}^{-2}, \qquad \|a_i^j\|_{S_i} = (a_i^j)^T S_i\, a_i^j, \qquad S_i = \begin{pmatrix} s_i^1 & & \\ & s_i^2 & \\ & & \ddots \end{pmatrix}$$

This defines the column version of the row-action method BICAV [10] mentioned in the previous section. Similarly as for the row version [8, Corollary 7.1] one finds that $\|A_i M_i A_i^T\|_2 \leq 1$ so convergence holds for $\omega_i \in (0, 2)$. In a similar way we can define column versions of DROP and SART.

To implement Cimmino (or point-SOR) we need to compute the 2-norms of all columns. This can be quite costly and inconvenient if the matrix A is not explicitly stored but instead available via functions that implement multiplication with A and its transpose. If we can assume that all matrix elements are nonnegative it is cheaper to compute the 1-norm since this can be done by performing a single matrix-vector product Ae with $e = (1, \ldots, 1)^T$. We then use for Cimmino

$$M_i = \begin{pmatrix} \|a_i^1\|_1 & & \\ & \|a_i^2\|_1 & \\ & & \ddots \end{pmatrix}^{-2}.$$

Since $\|a_i^j\|_2 \leq \|a_i^j\|_1$ it follows using (20) that again $\omega_i \in (0, 2)$ (without compromising convergence).

For point-SOR we take the scalars $M_i = 1/\|a_i\|_1^2$, $i = 1, 2, \ldots n$. As above we get

$$\rho\left(a_i M_i a_i^T\right) = 1/\|a_i\|_1^2\, \rho\left(a_i a_i^T\right) \leq 1/\|a_i\|_2^2 \rho(a_i a_i^T) = 1.$$

So again convergence is ensured for $\omega_i \in (0, 2)$. Normalization using the 1-norm is used, e.g., in [2], [23].

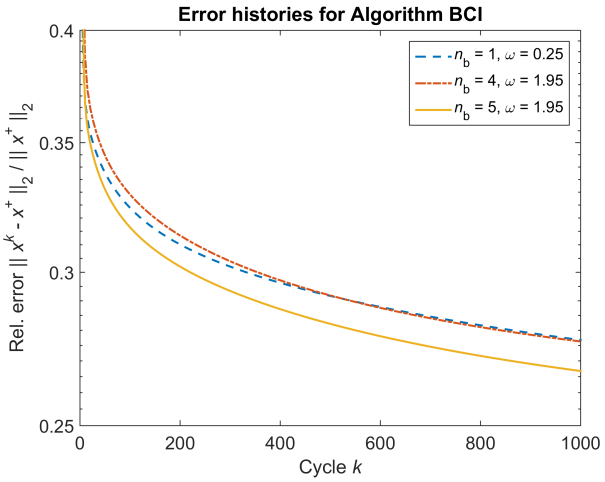

Fig. 1 Error histories for Algorithm BCI for different block sizes. Block size $n_{\rm b} = 5$ gives the fastest convergence.

3.4 A Numerical Example

We finish this section with a numerical example (in Matlab) that illustrates the convergence of Algorithm BCI for the **Cimmino**-case, cf. (19). The test problem is from parallel-beam computed tomography as implemented in the function `parallelbeam` from the package AIR Tools [17], and there is no noise in the data. The image is a 50×50 Shepp-Logan phantom, the detector has 71 pixels, and the projection angles are $5^\circ, 10^\circ, \ldots, 180^\circ$; hence the matrix A is $(36 \cdot 71) \times 50^2 = 2556 \times 2500$. All blocks have the same size $n_i = n_{\rm b}$ and relaxation parameter $\omega_i = \omega$, and we used the following combinations:

$$\begin{aligned} q &= 2500, \;\; n_{\rm b} = 1, \;\; \omega = 0.25, \\ q &= \;\; 625, \;\; n_{\rm b} = 4, \;\; \omega = 1.95, \\ q &= \;\; 500, \;\; n_{\rm b} = 5, \;\; \omega = 1.95, \end{aligned}$$

where ω was chosen experimentally to give fast convergence in each case.

The error histories of $\|x^k - x^+\|_2 / \|x^+\|_2$ versus k (where x^+ is the least squares solution) are shown in Fig. 1, and we see that for a well-chosen block size (which is problem dependent, here $n_b = 5$) we have faster convergence than for the point case $n_b = 1$. When implemented in a programming language that can efficiently utilize block operations, a block cycle is faster than a point cycle and the advantage of the block version becomes even more pronounced.

4 Implementation and Analysis of a Loping/Flagging Strategy

Haltmeier [16] discussed a "block-Kaczmarz" algorithm assuming that the system (1) is consistent. An interesting feature of his algorithm is the use of a *loping* strategy which – in terms of Algorithm BRI – omits the updating step associated with block

i if the residual norm $\|M_i^{1/2}(b_i - B_i z^{i-1})\|_2$ is below a certain threshold that reflects the noise in the data. The key idea is to use this as a stopping rule: once all updating steps of a cycle are omitted, the algorithm stops. While not discussed in [16], loping saves some computational work for blocks that are not updated; but note that we still need to compute the residual in order to decide whether to lope in block i.

4.1 Loping and Flagging in Algorithm BCI

A similar loping strategy can be introduced in Algorithm BCI, where we can choose not to update the solution block x_i^k if the associated vector $d_i^k = \omega_i M_i A_i^T r^{k,i}$ has a small norm. Again this will save computational work for blocks that are not updated. The Loping version of the algorithm takes the following form, where τ is a user-specified threshold:

Algorithm BCI-L: Block-Column Iteration – Loping Version

Initialization: $x^0 \in \mathbb{R}^n$ is arbitrary; $r^{0,1} = b - A\,x^0$.
For $k = 1, 2, 3, \ldots$ (cycles or outer iterations)
  For $i = 1, 2, \ldots, q$ (inner iterations)
    $d_i^k = \omega_i M_i A_i^T r^{k,i}$
    If $\|d_i^k\|_2 > \tau$
      $x_i^{k+1} = x_i^k + d_i^k$
      $r^{k,i+1} = r^{k,i} - A_i(x_i^{k+1} - x_i^k)$
    End
  End
  $r^{k+1,1} = r^{k,q+1}$
End

We now propose an *alternative strategy* that can potentially save much more computational work. Our key observation is that the situation $\|d_i^k\|_2 < \tau$ typically occurs when the associated solution block x_i has (almost) converged and very small updates are performed – while, at the same time, other blocks still have larger updates. Hence, we can choose to **flag** the ith block and not update it over the next N_{flag} cycles – without computing and checking $\|d_i^k\|_2$ and thus saving more work. In situations where large parts of the image converge fast, this can potentially save a lot of computational work without slowing down the convergence too much. The Flagging version of the algorithm takes the following form, where again τ is a user-specified thereshold:

Algorithm BCI-F: Block-Column Iteration – Flagging Version

Initialization: $x^0 \in \mathbb{R}^n$ is arbitrary; $r^{0,1} = b - A\,x^0$.
For $k = 1, 2, 3, \ldots$ (cycles or outer iterations)
  For $i = 1, 2, \ldots, q$ (inner iterations)
    If block-i is not flagged
      $d_i^k = \omega_i M_i A_i^T r^{k,i}$
      If $\|d_i^k\|_2 > \tau$
        $x_i^{k+1} = x_i^k + d_i^k$
        $r^{k,i+1} = r^{k,i} - A_i(x_i^{k+1} - x_i^k)$
      Else
        Flag block-i

```
            End
        Else
            If block-i has been flagged for N_flag outer iterations
                Unflag block-i
            End
        End
    End
    r^{k+1,1} = r^{k,q+1}
End
```

4.2 Convergence Analysis of Loping/Flagging

Note that loping/flagging can be seen as using non-stationary relaxation parameters,

$$\omega_i = \omega_i(k), \qquad i = 1, \ldots, q,$$

and also allowing some of these to be zero. The preceding convergence results are therefore not applicable. We will now analyze the convergence properties of loping/flagging for the SOR-case. The idea is to use the fact that the residual updates can be seen as coming from a projection method (as observed in [13, (2.5)]). In fact it follows from (8) that $r^{k,i+1} = (I - \omega_i A_i M_i A_i^T) r^{k,i}$. Hence with $M_i = (A_i^T A_i)^\dagger$, and using $(A_i^T A_i)^\dagger A_i^T = A_i^\dagger$ we get

$$r^{k,i+1} = \big(I - \omega_i A_i A_i^\dagger\big)\, r^{k,i} = \big(I - \omega_i P_{\mathcal{R}(A_i)}\big) r^{k,i}.$$

It follows that

$$\begin{aligned} r^{k,i+1} &= (1-\omega_i) r^{k,i} + \omega_i \big(I - P_{\mathcal{R}(A_i)}\big)\, r^{k,i} \\ &= (1-\omega_i) r^{k,i} + \omega_i P_{\mathcal{N}(A_i^T)} r^{k,i} = r^{k,i} + \omega_i \big(P_{\mathcal{N}(A_i^T)} r^{k,i} - r^{k,i}\big). \end{aligned}$$

We now allow non-stationary relaxation parameters and write the iteration as

$$r^{k,i+1} = r^{k,i} + \omega_i(k) \big(P_{\mathcal{N}(A_i^T)} r^{k,i} - r^{k,i}\big). \tag{21}$$

We next briefly describe the block-iterative projection (BIP) algorithm of Aharoni and Censor [1] for finding a point in the intersection $\mathcal{Q} = \cap_{i=1}^q \mathcal{Q}_i$, where $\mathcal{Q}_i \subset \mathbb{R}^n$, $i = 1, 2, \ldots, q$ is a collection of closed convex sets. Let

$$\omega^k = (\omega_1(k), \omega_2(k), \ldots, \omega_q(k)),$$

and let

$$P(\omega^k) = \sum_{i=1}^q \omega_i(k) P_{\mathcal{Q}_i}, \qquad \omega_i(k) \geq 0, \qquad \sum_{i=1}^q \omega_i(k) = 1.$$

Then BIP takes the form (where u^k is an iteration vector)

$$u^{k+1} = u^k + \lambda_k \big(P(\omega^k) u^k - u^k\big), \qquad \lambda_k \in (0, 2). \tag{22}$$

A sequence of weight vectors is called *fair* if for every $i \in \{1, 2, \ldots, q\}$ there exist infinitely many values of k such that $\omega_i(k) > 0$. It is shown in [1] that the iterates $\{u^k\}$ of (22) converge to a point in $\cap_{i=1}^q \mathcal{Q}_i \neq \emptyset$ provided $\{\omega^k\}$ is a fair sequence. Letting

$\mathcal{Q}_i = \mathcal{N}(A_i^T)$ (so that $\mathcal{Q} = \mathcal{N}(A^T)$) it is easy to see that given any sequence $\{\omega^k\}$ in (21) the generated iterates could also been generated by (22).

We illustrate with a small example. Let $q = 3$ and $\omega_1(k) > 0$, $\omega_2(k) = 0$, $\omega_3(k) > 0$. This would produce the following iterates in (21): $(r^{k,2}, r^{k,2}, r^{k,4})$. By taking $\lambda_k = \omega_1(k)$, $\lambda_{k+1} = \omega_3(k)$, $\omega^k = (1,0,0)$, $\omega^{k+1} = (0,0,1)$, and using the same starting value $r^{k,1}$ we get $u^{k+1} = r^{k,2}$, $u^{k+2} = r^{k,4}$.

Proposition 4 *Assume that $\{\omega^k\}$ is fair. Then for the SOR-version of Algorithms BCI-F and BCI-L the following results hold: (i) $\lim A^T(b - A\,x^k) = 0$. (ii) If $\mathcal{N}(A) = \emptyset$ then $x^k \to x^+$, where x^+ is defined in (13).*

Proof Define

$$x^{k+1} = \begin{pmatrix} x_1^{k+1} \\ \vdots \\ x_q^{k+1} \end{pmatrix}.$$

Then by construction $r^{k,q+1} = b - A\,x^{k+1}$, and by comparing (21) and (22) we have

$$r^{k,q+1} \to r^* \in \mathcal{N}(A^T)$$

which implies (i), and assuming an empty null space of A (ii) follows.

Remark 2 If only (i) holds then the sequence $\{x^k\}$ is said to be quotient convergent. For a stationary iteration, quotient convergence and convergence are equivalent [5]. However, since loping/flagging constitutes a non-stationary iteration we cannot draw this conclusion here.

4.3 Numerical Examples

We now compare the loping and flagging versions of Algorithm BCI with the standard version using the same parallel-beam test problem as before. Here, the image is 75×75, the detector has 106 pixels, and the projection angles are $1^\circ, 2^\circ, \ldots, 180^\circ$, so the matrix A has size $m \times n = (180 \cdot 106) \times 75^2 =$19080 $\times$ 5625. Again there is no noise in the data.

All algorithms are run with block size 1 (the point versions) and with $M_i = \|a_i\|_2^{-2}$. In Algorithm BCI-F we use the parameter $N_{\text{flag}} = 50$ that controls how long a pixel stays flagged.

To illustrate the advantage of the flagging idea, we first use a simple phantom with a small white disk of radius 5 pixels, centered in an otherwise black image. We expect that the pixels of the large homogeneous background will be flagged after some cycles, while the pixels inside and close to the disk need to be updated in each cycles.

The results are shown in Fig. 2 for three values of the threshold $\tau = 10^{-4}$, 10^{-5}, and 10^{-6}. The left plots show the error histories, and we see that if the threshold τ is too large the then convergence of BCI-L and BCI-F is severely slowed down compared to BCI. The middle plots give more insight, as they show the accumulated amount of work in each algorithm as a function of the number of cycles. Here, a "work unit" is the number of flops involved in the basic operations, namely, either an inner product ("dot") or a vector update ("saxpy"), both involving $2n$ flops. We see that BCI-L and, in particular, BCI-F save a lot of work compared to BCI.

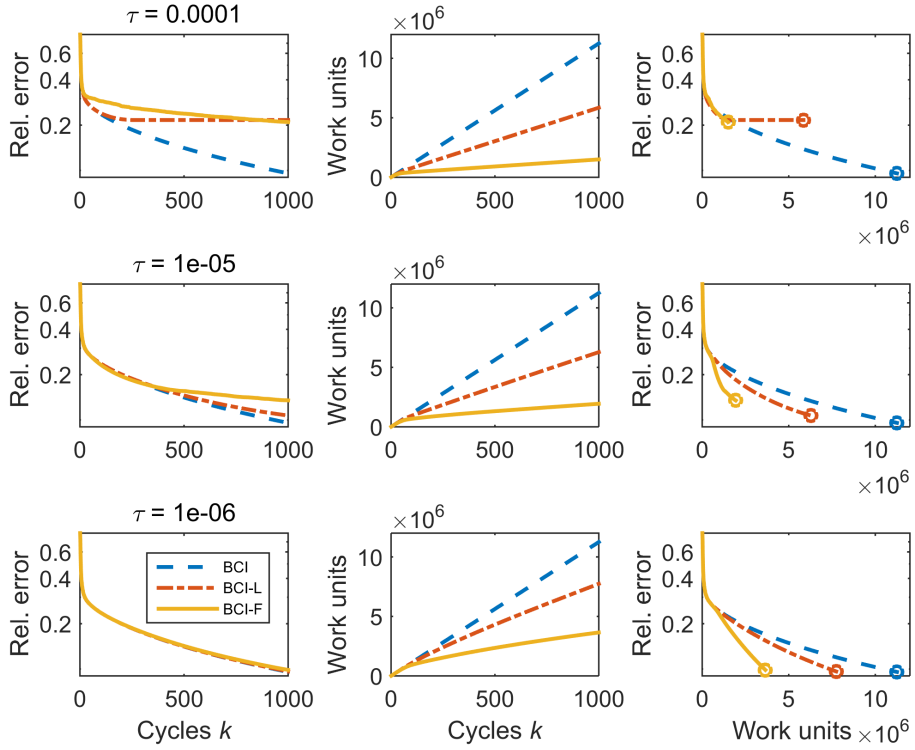

Fig. 2 Performance of Algorithms BCI, BCI-L and BCI-F for three values of the threshold τ used in loping and flagging. The left plots show the relative error $\|x^k - x^+\|_2/\|x^+\|_2$ versus the number of cycles k. The middle plots shows the accumulated amount of work as a function of the number of cycles. The right plots show the relative error versus the amount of work. A "work unit" is the number of flops involved in a inner product or a vector update.

The right plots, where we show the relative error versus the accumulated amount of work, provide an interesting perspective. We see that with a suitable size of the threshold (here, $\tau = 10^{-6}$), BCI-L and BCI-F reach the same accuracy as BCI with less work. In particular, BCI-F requires about 3 times less work than BCI to reach a relative error of about 0.1. We observe essentially the same behavior (not shown here) for experiments with the negative image where most pixels are white, suggesting that the behavior is essentially determined by the edges in the image.

To further illustrate the advantage of flagging, Fig. 3 shows the performance of BCI, BCI-L and BCI-F with $\tau = 10^{-6}$ applied to three random test images generated by means of the function `phantomgallery('ppower',75)` from AIR Tools [17]. This call generates the test images defined in [18] with large random regions of zeros and non-zeros. The behavior of the three algorithms is similar to the above and quite independent of the complexity of the image, and again BCI-F is the fastest algorithm. All our examples thus illustrate the potential advantage of incorporating flagging in the BCI algorithm.

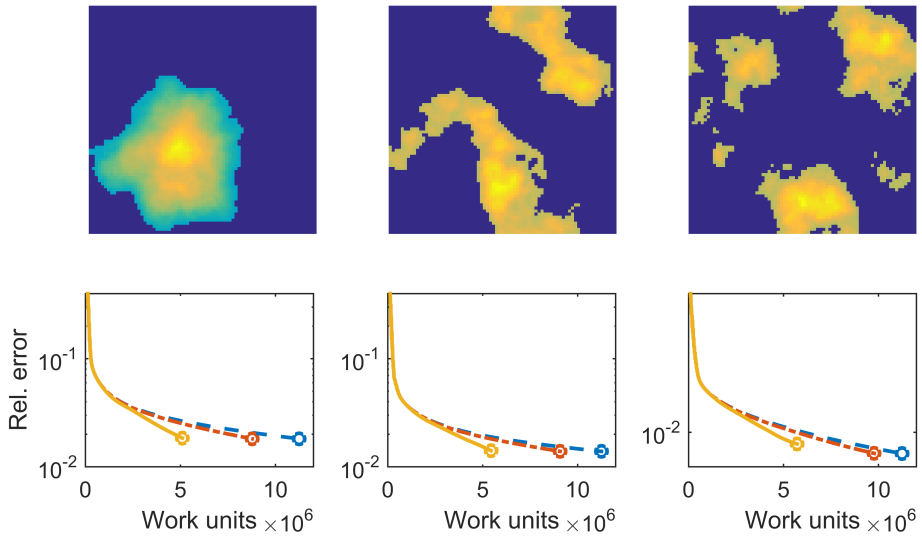

Fig. 3 Performance of Algorithms BCI, BCI-L and BCI-F with $\tau = 10^{-6}$. The top plots show three random test images generated with `phantomgallery('ppower',75)`, and the bottom plots show the corresponding relative error versus the amount of work. The overall behavior is quite independent of the complexity of the image.

5 Conclusion

We formulated algebraic (block) column-action methods in a common framework and proved that these methods – with different choices of weight matrices – converge to a least squares solution. We also introduced a "flagging" mechanism to save computational work by skipping small solution updates, and we proved some convergence results of the corresponding algorithm. We illustrated the numerical performance with examples from computed tomography showing the potential advantage of using blocks and flagging.

Conclusion

We thank the referee for reading the original manuscript carefully and for making several suggestions that improved the presentation.

Appendix A: Algorithm BCI as an Optimization Method

While we have treated Algorithm BCI as an algebraic method we want to emphasize that it can also be considered as a block coordinate descent optimization method for the unconstrained least squares problem

$$\min_x f(x), \qquad f(x) = 1/2\|A\,x - b\|_2^2 = 1/2(A\,x - b)^T(A\,x - b). \tag{23}$$

With a_i denoting the ith column of A, the partial derivatives are

$$\frac{\partial f}{\partial x_i} = a_i^T (A\,x - b), \qquad i = 1, \ldots, n. \tag{24}$$

Setting $\partial f / \partial x_i = 0$ leads to the expression

$$a_i^T\, a_i\, x_i = a_i^T \left(b - \sum_{j \neq i} a_j\, x_j \right)$$

and hence (with a bit of manipulation), from a given x we obtain the update of the ith component of x:

$$x_i^{\text{new}} = x_i + \frac{a_i^T}{\|a_i\|_2^2} (b - A\,x) \tag{25}$$

which is identical to the Cimmino and SOR point-versions of Algorithm BCI.

Along the same line, if we wish to minimize the objective function f with respect to all the variables associated with block column A_i then we set the corresponding partial derivatives to zero, which leads to the expression

$$A_i^T (A\,x - b) = 0 \quad \Leftrightarrow \quad A_i^T A_i\, x_i = A_i^T \left(b - \sum_{j \neq i} A_j\, x_j \right)$$

and we obtain the block updating

$$x_i^{\text{new}} = x_i + (A_i^T A_i)^{-1} A_i^T (b - A\,x) \tag{26}$$

which is identical to the SOR block-version of Algorithm BCI.

There is an alternative way to update the sub-vector x_i. Note that (25) can also be written as $x^{\text{new}} = x + \alpha_i\, e_i$ where $\alpha_i = a_i^T / \|a_i\|_2^2\, (b - A\,x)$ and e_i is the ith canonical unit vector. Then we can choose to compute a block update which is the mean of the updates for each component of x associated with block i:

$$x^{\text{new}} = x + \frac{1}{n_i} \sum_{j \in \mathcal{J}_i} \alpha_j e_j, \qquad \mathcal{J}_i = \text{column indices for block } i.$$

If a_i^j denotes the jth column of block A_i then such a block update takes the form

$$x_i^{\text{new}} = x_i + \frac{1}{n_i}\, \text{blockdiag}\left(\frac{1}{\|a_j^i\|_2^2} \right) A_i^T (b - A\,x). \tag{27}$$

This is identical to the Cimmino block-version of Algorithm BCI.

Appendix B: The Columns of the CT System Matrix

Recall that in computed tomography (CT) the columns of the matrix A (called the system matrix) are the forward projections of single pixels in the image. To understand these columns, we will briefly study the underlying Radon transform, associated with 2D parallel-beam problems, of delta functions in the image domain.

Recall that any line in a x, y-coordinate system can characterized by the relation $x\cos\theta + y\sin\theta = s$, where $\theta \in [0, \pi)$ is an angle and s is a translation parameter. Also recall that the Radon transform of a function $f(x, y)$ produces another function $g(s, \theta)$, called the sinogram, given by

$$g(s,\theta) = \int_{\text{image}} \delta(s - x\cos\theta - y\sin\theta)\, f(x,y)\, dx\, dy.$$

Hence the image of a delta function $\delta(x - x_\ell, y - y_\ell)$ located at (x_ℓ, y_ℓ) is characterized by the points in the sinogram that satisfy

$$s = x_\ell \cos\theta + y_\ell \sin\theta$$

which clearly describes a sine function (hence the name "sinogram"). Let us now consider two distinct points (x_1, y_1) and (x_2, y_2) and determine when/if the corresponding sine functions intersect, i.e., when/if $x_1\cos\theta + y_1\sin\theta = x_2\cos\theta + y_2\sin\theta$.

- If $x_1 = x_2$ we require $(y_1 - y_2)\sin\theta = 0$; since $y_1 \neq y_2$ this happens when $\theta = 0$.
- If $y_1 = y_2$ we require $(x_1 - x_2)\cos\theta = 0$; since $x_1 \neq x_2$ this happens when $\theta = \pi/2$.
- If both $x_1 \neq x_2$ and $y_1 \neq y_2$ then we arrive at the relation

$$(y_1 - y_2)\tan\theta = x_1 - x_2 \qquad \Leftrightarrow \qquad \tan\theta = \frac{x_1 - x_2}{y_1 - y_2}$$

  which always has a solution $\theta \in [0, \pi)$.

Hence we have shown that the two sine functions in the sinogram associated with two distinct points always intersect. For this reason, it is very likely that two distinct columns of A have at least one nonzero with the same row index.

Another way to see this is to note that each row of A is associated with a single X-ray passing through the image. A row of A has nonzero elements for those column indices for which the corresponding pixels are penetrated by the X-ray. With a large number of X-rays it is very likely that any given pair of columns of A will have at least one nonzero element for the same row index.

References

1. Aharoni, R., Censor, Y.: Block-iterative projection methods for parallel computation of solutions to convex feasibility problems. Lin. Alg. Appl., **120**, 165–175 (1989).
2. Andersen, A.H., Kak, A.C.: Simultaneous algebraic reconstruction technique (SART): A superior implementation of the art algorithm. Ultrasonic Imaging, **76**, 81–94 (1984).
3. Bai, Z.-Z., Jin C.-H.: Column-decomposed relaxation methods for the overdetermined systems of linear equations. Int. J. Appl. Math., **13**(1), 71–82 (2003).
4. Björck, Å., Elfving, T.: Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. BIT, **19**(2), 145–163 (1979).
5. Cao, Z.H.: On the convergence of iterative methods for solving singular linear systems. J. Comp. Appl. Math., **145**(1), 1–9 (2002).

6. Censor, Y.: Row-action methods for huge and sparse systems and their applications. SIAM Review, **23**, 444–466 (1981).
7. Censor, Y., Eggermont, P.P.B., Gordon, D.: Strong underregularization in Kaczmarz's method for inconsistent systems. Numer. Math., **41**(1), 83–92 (1983).
8. Censor, Y., Elfving, T.: Block-iterative algorithms with diagonally scaled oblique projections for the linear feasibility problem. SIAM J. Matrix Anal. Appl., **24**(1), 40–58 (2002).
9. Censor, Y., Elfving, T., Herman, G.T., Nikazad, T.: On diagonally relaxed orthogonal projection methods. SIAM J. Sci. Comput., **30**(1), 473–504 (2008).
10. Censor, Y., Gordon, D., Gordon, R.: BICAV: An inherently parallel algorithm for sparse systems with pixel-dependent weighting. IEEE Trans. Medical Imaging, **20**, 1050–1060 (2001).
11. Censor, Y., Zenios, S.A.: Parallel Optimization: Theory, Algorithms, and Applications. Oxford University Press, New York (1997).
12. Eggermont, P.P.B., Herman, G.T., Lent, A.: Iterative algorithms for large partitioned linear systems, with applications to image reconstruction. Lin. Alg. Appl., **40**, 37–67 (1981).
13. Elfving, T.: Block-iterative methods for consistent and inconsistent linear equations. Numer. Math., **35**(1), 1–12 (1980).
14. Elfving T., Nikazad, T.: Properties of a class of block-iterative methods. Inverse Problems, **25**(11), 115011 (2009).
15. Gordon, R., Bender, R., Herman, G.T.: Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. J. Theoretical Biology **29**(3), 471-81 (1970).
16. Haltmeier, M.: Convergence analysis of a block iterative version of the loping Landweber-Kaczmarz iteration. Nonlinear analysis, **71**, e2912–e2919 (2009).
17. Hansen, P.C., Saxild-Hansen, M.: AIR Tools – A MATLAB package of algebraic iterative reconstruction methods. J. Comp. Appl. Math., **236**(8), 2167–2178 (2012).
18. Jørgensen, J. S., Sidky, E. Y., Hansen, P. C., Pan, X.: Empirical average-case relation between undersampling and sparsity in X-ray CT. Inverse Problems and Imaging **9**(2), pp. 431–446 (2015).
19. Jiang, M., Wang, G.: Convergence studies on iterative algorithms for image reconstruction. IEEE Trans. Med. Imag., **22**(5), 569–579 (2003).
20. Keller, H.B.: On the solution of singular and semidefinite linear systems by iteration. J. Soc. Indus. Appl. Math. Ser. B, **2**, 281–290 (1965).
21. Kindermann, S., Leitão, A.: Convergence rates for Kaczmarz-type regularization methods. Inverse Probl. Imaging, **8**(1), 149–172 (2014).
22. Needell, D., Tropp, J.O.: Paved with good intentions: analysis of a randomized Kaczmarz method. Lin. Alg. Appl., **441**, 199–221 (2014).
23. Watt, D. W.: Column-relaxed algebraic reconstruction algorithm for tomography with noisy data. Applied Optics, **33**(20), 4420–4427 (1994).